WHAT IS CLAIMED IS:

1. A method for analyzing an object oriented program that supports dynamic class loading, the program comprising a plurality of objects, each object belonging to at least one class, the method comprising the steps of:

identifying a set A of classes in the program, wherein each of the classes within the set A is capable of, during execution of the program, causing loading of a class outside of the set A;

identifying a first set of method calls belonging to the classes in the set A that, during execution of the program, are capable of calling only methods belonging to a class within the set A;

identifying a second set of method calls belonging to the classes in the set A that, during execution of the program, are capable of calling methods belonging to a class outside the set A; and

storing data that identifies the first and the second set of method calls for subsequent use.

2. The method according to claim 1, further comprising the step of reporting the stored data to a user.

3. The method according to claim 1, further comprising the step of optimizing the program, based upon the stored data.

4. The method according to claim 3, further comprising the steps of:

for a given method call in the second set,

adding an optimization to the program that is valid when the given method call calls, during execution of the program, only methods belonging to a class in the set A; and

adding code to the program that triggers execution of the optimization when the given method call calls, during execution of the program, only methods belonging to the class in the set A.

5. The method according to claim 1, wherein said steps of identifying the first and the second set comprise the step of performing static analysis of the program.

6. The method according to claim 4, wherein said step of identifying the set A comprises the step of adding a bit in a class table corresponding to a given class and setting

the bit to a first predefined value, when the given class is within the set A.

7. The method according to claim 6, wherein said step of identifying the set A comprises the step of setting the bit to a second predefined value when the given class is newly loaded.

8. The method according to claim 7, wherein the code queries a class table corresponding to a runtime class of a reference variable associated with the given method call to determine whether the bit corresponds to the first or the second predefined value.

9. The method according to claim 1, wherein at least some of the classes in the set A are identified by a user.

10. The method according to claim 4, wherein the optimization corresponds to at least one of devirtualization and stack allocation.

11. The method according to claim 4, wherein the code is added to the given method call to enable optimization

with respect to the called methods.

12. A method for optimizing an object oriented program that supports dynamic class loading, the program comprising a plurality of objects, each object belonging to at least one class, the method comprising the steps of:

identifying a set A of classes in the program, wherein each of the classes within the set A is capable of, during execution of the program, causing loading of a class outside of the set A; and

adding an optimization to the program that is valid when a given method call belonging to the classes in the set A is capable of calling, during an execution of the program, only methods belonging to a class in the set A.

13. The method according to claim 12, further comprising the step of identifying the given method call based upon static analysis of the program.

14. The method according to claim 12, wherein the optimization corresponds to at least one of devirtualization and stack allocation.

15. A method for optimizing an object oriented program that supports dynamic class loading, the program comprising a plurality of objects, each object belonging to at least one class, the method comprising the steps of:

identifying a set A of classes in the program, wherein each of the classes within the set A is capable of, during execution of the program, causing loading of a class outside of the set A;

identifying a set B of method calls belonging to the classes in the set A that, during execution of the program, are capable of calling methods belonging to a class outside the set A;

for a given method call in the set B,

generating optimized code for replacing the given method call; and

generating test code that triggers execution of the optimized code in place of the given method call when the given method call calls, during execution of the program, only methods belong to a class within the set A.

16. The method according to claim 15, wherein said identifying steps are performed during an off-line phase.

17. The method according to claim 15, wherein said step of generating the optimized code further comprises the step of performing parametric data flow analysis on the program.

18. The method according to claim 15, wherein the optimization corresponds to at least one of devirtualization and stack allocation.

19. The method according to claim 15, wherein said steps of identifying the set B comprises the step of computing at each program point whether a reference variable will always correspond to one of the classes in the set A.

20. The method according to claim 19, wherein said computing step comprises the steps of:

constructing a class inheritance graph and a method call graph for the program;

computing at each program point a set C of all compile-time objects to which the reference variable is capable of pointing;

determining whether any of the compile-time objects within the set C correspond to one of the classes not in the set A;

designating the reference variable as not corresponding to one of the classes in the set A, when a compile-time object within the set C does not correspond to one of the classes in the set A;

designating the reference variable as always corresponding to one of the classes in the set A, when each of the compile-time objects within the set C corresponds to one of the classes in the set A.

21. The method according to claim 20, wherein said step of identifying the set A comprises the step of adding a bit in a class table corresponding to a given class and setting the bit to a first predefined value, when the given class is within the set A.

22. The method according to claim 21, wherein said step of identifying the set A comprises the step of setting the bit to a second predefined value when the given class is newly loaded.

23. The method according to claim 22, wherein the test code queries a class table corresponding to a runtime class

of the reference variable associated with the given method call to determine whether the bit corresponds to the first or the second predefined value.

24. The method according to claim 15, wherein the test code is added to the given method call to enable optimization with respect to the called methods.

25. The method according to claim 20, wherein compile-time objects with an identical runtime type are a same compile-time object.

26 A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for analyzing an object oriented program that supports dynamic class loading, the program comprising a plurality of objects, each object belonging to at least one class, said method steps comprising:

identifying a set A of classes in the program, wherein each of the classes within the set A is capable of, during execution of the program, causing loading of a class outside of the set A;

identifying a first set of method calls belonging to the classes in the set A that, during execution of the program, are capable of calling only methods belonging to a class within the set A;

identifying a second set of method calls belonging to the classes in the set A that, during execution of the program, are capable of calling methods belonging to a class outside the set A; and

storing data that identifies the first and the second set of method calls for subsequent use.

27 The method according to claim 26 further comprising the step of optimizing the program, based upon the stored data.

28 The method according to claim 25 further comprising the steps of:

for a given method call in the second set,

adding an optimization to the program that is valid when the given method call calls, during execution of the program, only methods belonging to a class in the set A; and

adding code to the program that triggers execution of the optimization when the given method call calls, during execution of the program, only methods belonging to a class in the set A.

29 A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for optimizing an object oriented program that supports dynamic class loading, the program comprising a plurality of objects, each object belonging to at least one class, said method steps comprising:

identifying a set A of classes in the program, wherein each of the classes within the set A is capable of, during execution of the program, causing loading of a class outside of the set A;

identifying a set B of method calls belonging to the classes in the set A that, during execution of the program, are capable of calling methods belonging to a class outside the set A;

for a given method call in the set B,

generating optimized code for replacing the given method call; and

generating test code that triggers execution of the optimized code in place of the given method call when the given method call calls, during execution of the program, only methods belong to a class within the set A.